Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5829–5837
Marseille, 11–16 May 2020
© European Language Resources Association (ELRA), licensed under CC-BY-NC

Dataset and Enhanced Model for Eligibility Criteria-to-SQL Semantic Parsing

Xiaojing Yu[1], Tianlong Chen[1], Zhengjie Yu[2],
Huiyu Li[3], Yang Yang[4], Xiaoqian Jiang[5], Anxiao Jiang[1]
[1] Texas A&M University, [2] University of Science and Technology of China, [3] UT Southwestern Medical Center,
[4] Walmart Technology, [5] University of Texas Health Science Center at Houston
{vicky_yu, wiwjp619}@tamu.edu, yzjfree@mail.ustc.edu.cn, huiyu.li@utsouthwestern.edu,
yang.yang2@walmart.com, xiaoqian.jiang@uth.tmc.edu, ajiang@cse.tamu.edu

Abstract

Clinical trials often require that patients meet eligibility criteria (e.g., have specific conditions) to ensure the safety and the effectiveness of studies. However, retrieving eligible patients for a trial from the electronic health record (EHR) database remains a challenging task for clinicians since it requires not only medical knowledge about eligibility criteria, but also an adequate understanding of structured query language (SQL). In this paper, we introduce a new dataset that includes the first-of-its-kind eligibility-criteria corpus and the corresponding queries for criteria-to-sql (Criteria2SQL), a task translating the eligibility criteria to executable SQL queries. Compared to existing datasets, the queries in the dataset here are derived from the eligibility criteria of clinical trials and include *Order-sensitive, Counting-based, and Boolean-type* cases which are not seen before. In addition to the dataset, we propose a novel neural semantic parser as a strong baseline model. Extensive experiments show that the proposed parser outperforms existing state-of-the-art general-purpose text-to-sql models while highlighting the challenges presented by the new dataset. The uniqueness and the diversity of the dataset leave a lot of research opportunities for future improvement.

Keywords: Semantic Parsing, Text-to-SQL, Eligibility Criteria

1. Introduction

In a clinical trial, eligibility criteria are guidelines for enrolling patients for a specific study. These guidelines decide whether a patient is eligible to participate in the study. Cohort definition, i.e., retrieving patients from an Electronic Health Records (EHR) database based on the eligibility criteria manually is a very time-consuming process for the clinical research. Semantically parsing eligibility criteria and translating them to a structured query language (i.e., criteria-to-sql) automatically can accelerate the cohort definition process significantly. However, the task is highly challenging and requires substantial domain-specific knowledge. Given an EHR database of patients and the eligibility criteria for a specific clinical trial, it aims at generating the corresponding SQL statements and retrieving eligible patients from the EHR database effectively.

Although there have been many works on generating SQL queries from unstructured texts (i.e., text-to-sql) that achieve nice performance on large-scale datasets such as WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018b), it remains difficult to learn a semantic parser directly from medical eligibility criteria for the criteria-to-sql task. First, to the best of our knowledge, there is currently no medical eligibility-criteria dataset publicly available for directly training an end-to-end semantic parser. Furthermore, the techniques derived from general natural-language-to-sql (NL2SQL) datasets such as WikiSQL do not transfer well to medical eligibility criteria due to the specific requirements in the medical domain. Sketch-based text-to-sql models have been widely studied in recent years. They take SQL structural information as prior knowledge to design slots for condition parts in SQL statements. Then they build neural semantic parsers to predict the value for each pre-defined slot. However, the sketch-based models built for existing NL2SQL datasets are not suitable for parsing

Patient Record Table

id	age	hypoperfusion	sBP	...	lactate_level
1	12	0	92	...	1.8
2	32	1	97	...	2.7
3	70	0	100	...	4.0

Eligibility Criteria

Evidence of refractory hypoperfusion attributed to sepsis (one or more of the following) : Systolic blood pressure less than 90 mmHg despite an intravenous fluid challenge of at least 30 ml/kg (a portion of this may be albumin equivalent) ; Blood lactate level at least 4 mmol/L.

SQL

SELECT id FROM table WHERE hypoperfusion = 1 and (sepsis = 1 or (case when sBP < 90 then 1 else 0 end + case when lactate_level > 4 then 1 else 0 end) >= 1)

Figure 1: An illustrative example from the Criteria2SQL dataset. In **eligibility criteria**, the concepts in red are column names in database tables. The values in blue are condition values. The phrase in green indicates a counting-based condition. In **SQL**, the conditions in the yellow background are Boolean-type conditions for which the condition value cannot be found in eligibility criteria, and the conditions in the blue background are position-type conditions for which the condition value can be found in eligibility criteria. Nested brackets are used to generate order-sensitive conditions from the eligibility criteria.

medical eligibility criteria in the following cases:

- **Order-sensitive eligibility criteria:** Eligibility criteria can include multiple conditions with mixed AND/OR operators. In this case, the existing mod-

els can fail to predict the precedence of operators on conditions, and lead to semantic ambiguities. For instance, given an eligibility criterion "*Patients have a known or suspected infection and meet criteria for severe sepsis or septic shock*", two possible SQL queries "infection = 1 and sepsis = 1 or septic_shock = 1" and "infection = 1 and (sepsis = 1 or septic_shock = 1)" return different results. Therefore, grouping conditions with brackets in the correct order is important for guaranteeing the precedence of AND/OR operators in SQL queries.

- **Counting-based eligibility criteria:** Eligibility criteria are often long free texts with multiple conditions but require patients to satisfy only a part of them. The existing text-to-sql models have ignored this case. However, it is a common situation for eligibility criteria in clinical trials, such as "*The presence of at least two of the following criteria: (a) body temperature >38; (b) pulse rate >90/ min; (c) breath rate >20/min;*". The correct SQL query is "(case when temperature >38 then 1 else 0 end + case when heart_rate >90 then 1 else 0 end + case when respiratory_rate >20 then 1 else 0 end) >= 2 ".

- **Boolean-type eligibility criteria:** A widely adopted assumption in existing table-aware semantic parsing algorithms is that every condition value should appear in the input utterance. Therefore the models are supposed to generate the condition value by predicting its position index and extracting it from the utterance. However, this assumption often does not hold in medical eligibility criteria. When a criterion requires a true-or-false statement for disease diagnosis, the corresponding condition value is Boolean and is often implicitly stated in the criterion. An example is the SQL query "bleeding = 1" for the corresponding eligibility criterion "*active bleeding*".

We illustrate the above three cases with an example in Figure 1. Such examples are very common in clinical trials, but are rarely studied in the NL2SQL domain. In this paper, we address these challenging cases.

To fill the gap between open-domain text-to-sql tasks and the clinical criteria-to-sql tasks, we collect and annotate the first-of-its-kind eligibility-criteria dataset (Criteria2SQL) for learning semantic parsing of eligibility criteria. The criteria in the dataset are collected from clinical trials for Sepsis, Heart attack, Diabetes and Alzheimer's diseases. The dataset contains 2003 eligibility criteria with corresponding SQL queries, and covers 984 concepts. The criteria have different levels of complexity, and cover (but are not limited to) the three cases mentioned above.

To translate criterion to SQL, we design a context-free grammar for generating SQL statements. We also present a strong model based on the grammar. The contributions of the paper include:

- We present a first-of-its-kind dataset for clinical eligibility-criteria-to-sql tasks, which covers (but is not limited to) *Order-sensitive, Counting-based, and Boolean-type* criteria. The complexity of the eligibility criteria and the corresponding SQL annotations reflect the real challenges for NL2SQL tasks in the clinical domain.

- To handle the three challenging cases, we propose a new context-free grammar for generating SQL queries for the eligibility criteria. Moreover, new types of slots for the sketch-based semantic parsing are designed based on the grammar.

- We design a strong baseline model to semantically parse eligibility criteria. Extensive experiments show the model's superior performance compared to existing general-purpose models.

2. Related Work

Our study is closely related to two research areas: semantic parsing and formalization of eligibility criteria.

Semantic parsing: Semantic parsing maps a natural language to its structured meaning representation, such as logical forms (Dong and Lapata, 2016; Reddy et al., 2016; Le and Zuidema, 2012), executable programs (Hayati et al., 2018; Iyer et al., 2018), arithmetic equations (Kushman et al., 2014; Mehta et al., 2017) and database queries (Pasupat and Liang, 2015; Yin et al., 2016). There has been study on semantic parsing for generating structured query language (SQL) from unstructured questions (Iyer et al., 2017; Xu et al., 2017; Yu et al., 2018a; Dong and Lapata, 2018; Hwang et al., 2019; Lin et al., 2019; Guo et al., 2019), which has achieved nice performance on WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018b). However, existing works mostly did not handle complex combinations of conditions in SQL's ubiquitous *WHERE* clause, including the *order-sensitive, counting-based and Boolean-type* cases discussed above. Handling such complex criteria and queries is important for clinical trials.

Formalization of eligibility criteria: Extensive study has focused on turning eligibility criteria to structured representations such as SQL. (Lonsdale et al., 2008; Ross et al., 2010; Tu et al., 2011; Luo et al., 2011; Milian and ten Teije, 2013). The work (Wang et al., 2019) proposes the first text-to-SQL based medical QA dataset, while the synthesized questions are very different from clinical eligibility criteria in both the sequence lengths and their content. Some studies(Levy-fix et al., 2015; Kang et al., 2017; Yuan et al., 2019) extract information or structure eligibility criteria into other intermediate representation based on the OMOP Common Data Model(Hripcsak et al., 2015). Most existing eligibility-criteria filters use relatively simple templates to generate SQL queries, which often cannot handle nested conditions or complex combinations of conditions.

3. SQL Generation from Criteria

The Criteria2SQL dataset we present contains EHR tables, eligibility criteria and their SQL annotations. In this section, we first use a specific example to illustrate these components and our task in detail. Then, we describe the dataset collection process. Finally, we analyze the collected dataset and provide its statistical properties.

Property	Example Annotation	stats.
Position-Type Condition	Query: At least 18 years of age SQL: age >= 18	52.3 %
Boolean-Type Condition	Query: Pregnancy SQL: pregnancy_or_lactation = 1	52.4 %
Order-Insensitive Condition (Multi-Condition without Brackets)	Query: Patients fulfilling the criteria of severe sepsis or septic shock SQL: severe_sepsis = 1 or septic_shock = 1	38.1 %
Order-Sensitive Condition (Nested Condition with Brackets)	Query: baseline hgb <11.7 g / dl in female , or <13 g / dl in male SQL: (hgb <11.7 and gender = 'female') or (hgb <13 and gender = 'male')	13.9 %
Counting-based Condition	Query: At least two of the following: low blood pressure prior to volume expansion (sbp <100) ; glasgow coma scale <13 ; skin mottling score >2 SQL: (case when sbp <100 then 1 else 0 end + case when glasgow_come_scale <13 then 1 else 0 end + case when skin_mottling_score >2 then 1 else 0 end) >= 2	1.2 %

Table 1: Properties of eligibility criteria and their pairing SQL annotations in our dataset. Here 'position-type conditions' and 'multi-conditions without brackets' are common conditions that also exist in other open-domain datasets, while the other three types of conditions are unique properties only considered in our dataset.

3.1. Task Description

For the eligibility criteria given in Figure 1:
"Evidence of refractory hypoperfusion attributed to sepsis (one or more of the following): systolic blood pressure less than 90 mmHg despite an intravenous fluid challenge of at least 30 ml/kg (a portion of this may be albumin equivalent); Blood lactate level at least 4 mmol/L. "
we want to select eligible patients from an EHR table (which table structure is as shown in Figure 1) using a model-generated SQL query as follows:
"SELECT id FROM table WHERE **hypoperfusion** = 1 and (**sepsis** = 1 or (case when **sBP** <90 then 1 else 0 end + case when **lactate_level**>= 4 then 1 else 0 end) >= 1)".
The task we study is to train a model and use it to generate the SQL query for each given eligibility criteria. We will present a model that firstly predicts the column names (following the example above) by matching medical concepts *hypoperfusion*, *sepsis*, *systolic blood pressure* and *lactate level* in the eligibility criteria with the corresponding column names in the table, then determines the corresponding condition operator and the condition value. The tuple (column name, condition operator, condition value) is treated as a single condition. The model will concatenate conditions with AND/OR operators in a correct execution order by adding nested brackets. Counting-based eligibility criteria form a special family of conditions, which require a '**case when**' statement with the summation operator to count the number of satisfied conditions.

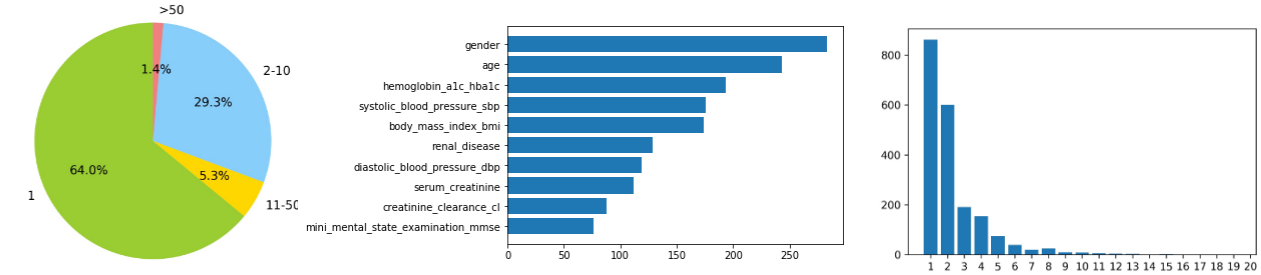
3.2. Dataset Collection

In order to train the model, i.e., a semantic parser to perform the task described above, we need to create a dataset that includes patient EHR tables, eligibility criteria, and corresponding SQL queries. The data collection process is as follows:

Data source. To collect a large number of eligibility criteria, we have used the clinical trials registered in Clinicaltrials.gov, and crawled eligibility criteria from the website with the keywords *Sepsis*, *Heart attack*, *Diabetes* and *Alzheimer's*. After obtaining the raw eligibility criteria, we have performed preprocessing steps to make the data suitable for training a neural semantic parser. We have removed incomplete and ambiguous sentences to make the criteria logically explicit. Then we have removed sentences containing only temporal information, time-dependent attributes or subjective judgments of physicians to make sure the processed criteria are compatible with EHR tables. We have obtained 1983 qualified criteria for SQL annotation. The dataset also includes 20 additional very long eligibility criteria from other clinical trials to increase the coverage on counting-based cases. In total, our dataset includes 2003 criteria and their corresponding SQL annotations.

Concept set. A concept set has been extracted from eligibility criteria and used for generating column names for synthetic patient-record tables. The concept set is also used for SQL annotations. In order to obtain a high-quality concept set, a medical expert has labeled medical concepts and demographics for each eligibility criterion, and collected the concepts to form an initial concept set. Since abbreviations and synonyms of medical concepts are widely used, similar concepts have been merged. The finalized concept set includes 984 concepts. The value types (i.e., *bool*, *int*, *float*, and *string*) and the valid range of values for each concept have been recorded for the synthesis of EHR tables.

SQL annotation. The SQL annotations have been created by three SQL experts. Because these SQL annotations are used to select patients who meet the eligibility criteria, they all share the same structure:
SELECT *patient_id* FROM *table_id* WHERE *condition*.
Annotators have been required to fill in the *condition* part of the *WHERE* clause. In order to reduce the annotators' workload, the column names remained the same as the terms used in the eligibility criteria instead of selecting them from the concept set. After the manual labeling phase finished, the column names were replaced by the concepts from the concept set by using a lookup table created by the



(a). Number of column name occurrences (b). Number of Top-10 column name occurrences (c). Distribution of condition number

Figure 2: Statistics of the Criteria2SQL dataset.

concept-set construction phase.

Conditions and consistency. The annotated SQL queries include the following cases: single condition, conditions combined by AND/OR operators without brackets, nested conditions combine by AND/OR operators and brackets, and counting-based conditions. Table 1 shows examples of the eligibility criteria and their SQL annotations. When an eligibility criterion includes multiple conditions, it may have multiple valid SQL queries. We require the order of the conditions in the SQL query to be consistent with the order of the corresponding concepts in the criterion. Each generated SQL annotation has been verified by the two other annotators to ensure the meaning of SQL query is consistent with the text. Then we double-checked the annotation by executing the SQL query on the synthetic EHR table.

EHR database synthesis. Since real EHR data are protected by law due to privacy concerns, we have synthesized the EHR tables for the collected eligibility criteria and stored them in the database. We first shuffle all the SQL queries, and then use every 5 queries as a group to generate a table. The column names from the SQL-query group are used as the column names of the generated table.

3.3. Dataset Analysis

To illustrate the properties of the Criteria2SQL dataset, we analyze the distribution of concepts in the dataset, the complexity of SQL queries, and compare our dataset with the existing text-to-sql datasets.

Concept distribution analysis. The dataset contains 984 concepts in 2003 queries, which shows the diversity of the medical concepts. The number of occurrences of each concept in the SQL queries is illustrated in Figure 2 (a). 64.0% of the concepts appear only once, indicating the natural sparsity of concepts from different clinical trials. 29.3% of the concepts appear 2 to 10 times. 1.4% of the concepts appear more than 50 times. Figure 2 (b) lists the occurrences of top-10 concepts. Demographic concepts in the eligibility criteria such as age and gender appear most frequently. Based on the statistics as shown in Figure 2, most clinical trials require patients to satisfy a specific range requirement for age and have varied requirements (such as systolic blood pressure and diastolic blood pressure) according to their genders.

Complexity analysis. The complexity of the generated SQL queries is analyzed in both quantitative and qualitative ways. We count the number of single conditions in each SQL query (a.k.a. *condition number*). Figure 2 (c) presents the number of SQL queries with different condition numbers. 56.7% of all queries include at least 2 conditions and the longest query has 20 conditions, which could be very difficult to parse. We analyzed the distribution of three unique eligibility criteria (Order-sensitive condition, Counting-based condition, Boolean-type condition) as shown in Table 1. 52.4% of all queries include the Boolean-type condition and 52.3% of them include the position-type condition. 27% of all queries include at least 3 conditions, of which 70.2% are nested order-sensitive conditions. 1.2% of queries include counting-based conditions that usually exist in very long eligibility criteria. Hence, the specific design for parsing the three particular types of conditions and a combination of them are required in the criteria-to-sql task.

Dataset	Criteria2SQL	GeoQuery	ATIS	WikiSQL	Spider
# of Queries	2003	877	5280	80654	10181
# of Tables/DB	1	6	32	1	5.1
# of CASE WHEN	24	0	0	0	0
# of Nested Conditions	379	0	0	0	0
# of Implicit Boolean Condition Value	1050	0	0	0	0

Table 2: Comparison of datasets

Comparison with other datasets. We compare our dataset with the other text-to-sql datasets, and show the results in Table 3.3. Our dataset is a medium-scale dataset, but it is the first dataset that considers the execution priority of conditions and includes nested parenthesis in SQL queries. The newly added SQL patterns enable the learning of predictions based on counting-based conditions, combination of multiple conditions, and implicit Boolean condition values. For very popular datasets such as WikiSQL and Spider, the questions were generated from given tables. In our dataset, we focus on parsing the distinct expressions of the eligibility criteria from real clinical trials and the SQL generation over multiple tables is not our concern.

4. Grammar-based Baseline Model

Sketch-based semantic parsers adopt SQL structures as prior knowledge to design slots for different SQL components, and convert a text-to-sql task to a slot-filling task (Dong and Lapata, 2018; Xu et al., 2017). Grammar-based neural models have shown their effectiveness in complex SQL generation (Guo et al., 2019; Lin et al., 2019). We apply a sketch-based neural approach to generate the components in SQL queries. To utilize the grammar-based method

in our task, we define a context-free grammar for SQL generation in the criteria-to-sql task, and design slots under the constraints of the new grammar rules.

4.1. SQL Grammar

We summarize the SQL queries in our dataset, and derive context-free grammar rules from them as follows:

(1) $SQL \rightarrow A\ S$
(2) $A \rightarrow SELECT\ id\ FROM\ table\ WHERE$
(3) $S \rightarrow B \mid C \mid S\ o_1\ S \mid (S\ o_1\ S)$
(4) $B \rightarrow (a\ T\ b\ o_2\ C_v)$
(5) $C \rightarrow C_n\ o_2\ C_v$
(6) $T \rightarrow S\ b\ o_3\ a\ S \mid T\ b\ o_3\ a\ S$
(7) $o_1 \rightarrow AND \mid OR$
(8) $o_2 \rightarrow = \mid <> \mid > \mid >= \mid < \mid <=$
(9) $o_3 \rightarrow +$
(10) $a \rightarrow (CASE\ WHEN$
(11) $b \rightarrow THEN\ 1\ ELSE\ 0\ END)$

Here S denotes conditions in the *WHERE* clause. C and B represent the single conditions and the counting-based conditions, respectively. C_n denotes the column names and C_v denotes the condition values. o_1 denotes the combination operators AND/OR, o_2 denotes the comparison operators, and o_3 denotes the summation operator. An illustrative parse tree is shown in Figure 3(b).

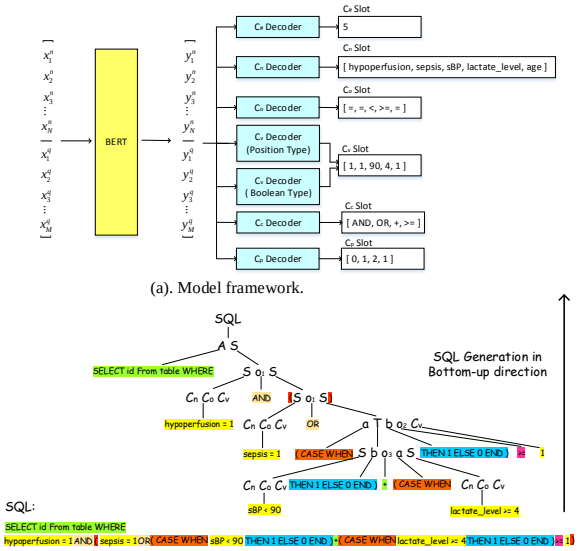
Rule (1) generates a complete SQL query. Rule (2) is used for the *SELECT* statement. In rule (3), by using (So_1S), we can generate the SQL components recursively by combining multiple conditions with the brackets. Rule (5) generates single conditions to compare the columns with the condition values. Rules (4,6,9-11) are designed for generating counting-based conditions, which are used for the comparison of the number of satisfied conditions with an integer C_v. In rules (4,10-11), aTb converts the True/False statement of the conditions to binary digits by using 'CASE WHEN' and 'IF ELSE' patterns, with '1' representing *True* and '0' representing *False*. In rule (6), the summation operator o_3 is used to sum up binary digits. Using the above grammar rules, we can generate SQL queries for very complex eligibility criteria with mixed counting-based conditions and order-sensitive conditions. For example, we can generate SQL queries with parts such as *'at least 3 of the following conditions should be satisfied: ...'*.

4.2. Slot Design

With the proposed grammar, our neural semantic parser predicts the structure of the *WHERE* clause and the value of slots $C_{\#}, C_n, C_o, C_c, C_p, C_v$ to generate the query. The slots are explained as follows:

- Condition number $C_{\#}$ slot: the number of single conditions in SQL query. The parser needs to decide how many single conditions exist in the query.
- Column name slots C_n: the column names in all single conditions. The parser needs to decide which columns to include in the *WHERE* clause.
- Comparison operator slots C_o: the comparison operators in all single conditions. The parser needs to choose the operator from the comparison-operator list $\{=, <>, >, >=, <, <=\}$.
- Combination operator slots C_c: the extended operators in conditions. The parser needs to choose the operator from $\{AND, OR, +, =, <>, >, >=, <, <=\}$.
- Combination priority slots C_p: the priority levels of all operators. The parser needs to decide the order to execute the operators.
- Condition value slots C_v: condition values in all single conditions. The parser needs to decide the values in the comparison conditions.

Similar concepts for $C_{\#}, C_n, C_o, C_v$ are proposed in SQL-Net (Xu et al., 2017) to construct simple SQL conditions. We propose additional operators to create the extended combination operator slots C_c and the priority slots C_p for generating complex SQL conditions. First, the parser predicts the value for $C_{\#}$ to decide the number of conditions in the query. Then the parser predicts C_n, C_o, C_v to generate simple conditions. The parser further combines the conditions by predicting the combination operators C_c and their corresponding priority value (i.e., C_p slots) to generate the final query.



(a). Model framework.

(b). The parse tree of SQL query in Figure 1.

Figure 3: Framework of the proposed neural parser. We feed the embeddings into separate bi-directional LSTM decoders to predict the values for each slot. Then values of the slots are used to generate the SQL query based on the proposed context-free grammar. Part (b) shows an example for generating the SQL query from the slots. Note that when the combination operator in C_c is '>=', based on rule (4), we only use C_v to generate SQL component; thus the corresponding C_n('age') and C_o('=') values are ignored.

4.3. Model

Given a table with N column names and the eligibility criteria as a text of length M, the input sequence for our model is defined as

$$x = [x_1^n; x_2^n; x_3^n; ...; x_N^n | x_1^q; x_2^q; x_3^q; ...; x_M^q]$$

where x_i^n is the i-th column name and x_j^q is the j-th word in the eligibility criteria. We use the technique in (Zhong et al., 2017) to process the column names of the table. Then we employ the pre-trained BERT model to encode the N column names and the M words in the eligibility criteria into word embedding. The output embedding sequence is denoted by

$$y = [y_1^n; y_2^n; y_3^n; ...; y_N^n | y_1^q; y_2^q; y_3^q; ...; y_M^q]$$

where y_i^n and y_j^q are 1×512 embedding vectors. We denote the embedding of the column-name list $[y_1^n; y_2^n; y_3^n; ...; y_N^n]$ by y_n and the embedding of the eligibility criteria $[y_1^q; y_2^q; y_3^q; ...; y_M^q]$ by y_q.

Figure 3 shows an example of the SQL query generated by the model. The network first adopts the pre-trained BERT model (Devlin et al., 2018) for the encoding of the one-hot input sequence into the word embedding sequence. It then feeds the word sequence to an LSTM Encoder-Decoder Model for the slot prediction.

With the slots defined above, the criteria-to-sql task is converted to classification tasks for the different slots. In our dataset, the maximum number of conditions is 20. Thus the decoder predicts the $C_\#$ slot value as a 20-way classification. The class numbers are N for C_n slot, 6 for C_o (i.e., the size of the comparison operator set), and 9 for C_c (i.e, the size of the combination operator set). We assume that the depth of the nested condition ranges from 0 to 4, so the class number for C_p is 5. The class numbers for the C_v slots, i.e., the position-type condition value and the Boolean-type condition value, are M and 2 , respectively.

We adopt the column attention mechanism from SQLNet (Xu et al., 2017) and the NL2SQL decoder from (Hwang et al., 2019) to build our model. The decoder is a two-layer bi-directional LSTM (Hochreiter and Schmidhuber, 1997) module without sharing the weights, and is used to predict the value of each slot separately. The probability calculation of the $C_\#, C_n, C_o$ slots is the same as in the SQLNet. And the softmax classifier is used to predict the output class.

The probabilities of the C_p, C_c slots are calculated as follows:

$$P_{c_p}(i|y_n) = softmax(W_1^{c_p} tanh(W_2^{c_p} y_q + W_3^{c_p}(H_q^{c_p} w^{c_p}))) \quad (1)$$

$$P_{c_c}(i|y_n) = softmax(W_1^{c_c} tanh(W_2^{c_c} y_q + W_3^{c_c}(H_q^{c_c} w^{c_c}))) \quad (2)$$

where $W_i, i \in \{1, 2, 3\}$ are the trainable weight matrices for the different decoders, H_q is the $512 \times M$ matrix with the i-th column representing the hidden-state output of the LSTM that has the i-th word in the eligibility-criteria text as the current input, and w is the $M \times 1$ attention weight vector of the criteria.

For C_v slots, since there are two types of condition values (the position-type and the Boolean-type), we construct decoders for both types. During training, the selection of the C_v decoder for generating the condition values depends on the value type of the predicted condition name. The inference of the condition value is as follows

$$P_{c_v}(i|y_n) = \mathbb{1}_{C_t=bool} P_{bool}(i|y_n) + (1 - \mathbb{1}_{C_t=bool}) P_{pos}(i|y_n) \quad (3)$$

$$\begin{aligned} P_{bool}(i|y_n) &= softmax(b(i|y_n)) \\ b(i|y_n) &= W_1^{bl} tanh(W_2^{bl} y_q + W_3^{bl}(H_q^{bool})) \end{aligned} \quad (4)$$

$$\begin{aligned} P_{pos}(i|y_n) &= softmax(p(i|y_n)) \\ p(i|y_n) &= W_1^{pos} tanh(W_2^{pos} y_q + W_3^{ps}(H_q^{ps} + W_3^{ps} h)) \end{aligned} \quad (5)$$

where $\mathbb{1}_{C_t=bool}$ is an indicator function that returns 1 when the value type of the predicted column name is Boolean, and returns 0 otherwise. P_{bool} and P_{pos} denote the probabilities of the Boolean-type decoder for the C_v slot and the position-type decoder for the C_v slot. Since the position-type decoder needs to predict the start index and the end index of the value in the eligibility-criteria text, $P_{pos}(i|y_n)$ is a 2×1 vector with elements $P_{pos}(i|y_n)^{start}$ and $P_{pos}(i|y_n)^{end}$.

Model	Dev SQL(%)	Dev X(%)	Test SQL(%)	Test X(%)
SQLNet(Xu et al., 2017)	9.33	10.33	13.20	13.86
SQLova(Hwang et al., 2019)	6.00	6.60	11.88	12.87
Ours	16.67	20.67	14.20	15.84

Table 3: Overall accuracy results. Here Dev SQL, Dev X, Test SQL and Test X stand for the SQL accuracy for development set, execution accuracy for development set, SQL accuracy for test set and execution accuracy for test set, respectively.

Model	Set	$C_\#$	C_n	C_o	C_v(Boolean/Pos)	C_c	C_p
SQLNet	Dev	58.7	21.2	46.6	18.8/0.3	-	-
	Test	63.0	20.1	55.7	18.8/0	-	-
SQLova*	Dev	74.6	25.5	44.3	36.5/-	-	-
	Test	66.4	47.5	77.5	76.7/-	-	-
Ours	Dev	60.3	35.9	54.3	89.1/16.8	35.0	25.4
	Test	67.3	21.8	65.7	40.9/62.1	35.1	26.4

Table 4: Sub-module accuracy results for all methods. Here the sub-module accuracy of SQLova* is calculated as the accuracy for SQLova after first removing eligibility criteria with Boolean-type conditions.

5. Performance Evaluation

We have conducted extensive experiments on the Criteria2SQL dataset, and compared the performance of two advanced and general text-to-sql models in the literature to our baseline model. Code implementation and Criteria2SQL dataset can be found at: `github.com/xiaojingyu92/Criteria2SQL`

5.1. Experimental Setup

We partition the Criteria2SQL dataset into a training set (70%), a development set (15%) and a test set (15%) uniformly randomly. SQLNet (Xu et al., 2017) and SQLova (Hwang et al., 2019) with the NL2SQL decoder are used for comparison. Two metrics are used to evaluate the generated SQL queries, including *SQL accuracy* and *execution accuracy*. The SQL accuracy requires exact matching between the generated SQL queries and the ground truth. The execution accuracy compares the query results (i.e., selected patients) of running the generated SQL queries to the ground

Type	Set	# Query	# Conditions$_{/Query}$	SQL ACC	X ACC
Boolean-Type	Dev	48	[1,7]	35.42	35.42
	Test	71	[1,13]	11.27	11.27
Order-Sensitive	Dev	138	[3,16]	0.73	2.17
	Test	53	[3,17]	0	3.77
Counting-Based	Dev	4	[6,16]	0	0
	Test	4	[8,9]	0	0

Table 5: Statistics on unique conditions in Dev/Test Set. ('SQL ACC' and 'X ACC' mean SQL accuracy and execution accuracy.)

Type	Set	$C_{\#}$	C_n	C_o	C_v	C_c	C_p
Boolean-Type	Dev	56.3	33.7	75.0	65.2	21.7	16.3
	Test	70.4	10.7	82.1	61.9	35.7	25.6
Order-Sensitive	Dev	43.5	28.2	46.4	19.5	35.3	27.0
	Test	24.5	13.8	60.3	39.02	43.6	27.2
Counting-Based	Dev	0	7.5	27.5	12.5	22.5	10.0
	Test	0	8.6	60.0	48.6	8.6	40.0

Table 6: Sub-module accuracy results on unique conditions. It matches each predicted slot value from sub-modules with the corresponding value in the ground truth.

truth over EHR tables. Since the BERT model and the word embedding layers in SQLNet are pre-trained on an open-domain corpus with substantial difference from a clinical corpus, we have fine-tuned the embedding modules in all models during training.

5.2. Experimental Results

We compare our model to both SQLNet and SQLova. Their performance is presented in Table 4.3. Our model shows a significant improvement for the development set by 7.34% (for SQL accuracy) and 10.34% (for execution accuracy) compared to SQLNet, and by 10.67% (for SQL accuracy) and 14.07% (for execution accuracy) compared to SQLova. Similarly, the accuracy for the test set is also higher than both other models. Since SQLova relies on position-based conditions and cannot handle Boolean-type conditions, eligibility criteria with Boolean-type conditions are removed before feeding to the SQLova model. Only 134 (respectively, 89) eligibility criteria for the development set (respectively, test set) can generate SQL queries for the SQLova model, which results in the lowest overall accuracy.

To analyze the performance for each slot, we calculate the prediction accuracy for each slot category (called *sub-module accuracy*). See Table 4.3. Same as before, the sub-module accuracy for SQLova is calculated based on the 134(dev)/89(test) position-based eligibility criteria. Since SQLNet and SQLova do not have decoders for C_c and C_p slots, they cannot parse counting-based or order-sensitive conditions. Our model achieves higher accuracy in predicting condition values (including Boolean-type conditions and position-type conditions) on both dev set and test set. As shown in Table 5.1, 48(dev)/71(test) eligibility criteria include only Boolean-type conditions. The high accuracy in predicting Boolean-type conditions is one main reason for our model to achieve the highest overall accuracy.

5.3. Error Analysis

As shown in Table 4.3, the predictions for slots C_n and C_p are the bottleneck for the performance of our model. To investigate the model's performance for the three unique types of eligibility criteria, we analyze the distribution and accuracy for the Boolean-type, order-sensitive and counting-based conditions separately. Table 5.1 reveals that although our model can predict Boolean-type conditions relatively well, it fails to parse most order-sensitive conditions and all counting-based conditions. Most order-sensitive conditions and counting-based conditions have a long sequence of conditions, especially for counting-based conditions (at least 6/8 conditions in dev/test set). They make slot prediction very challenging for two reasons. First, it is difficult to predict the correct $C_{\#}$ for long sequences, as shown in Table 5.1. Second, although the prediction of each slot for a single condition could have high accuracy, it is very hard to predict all conditions correctly in a long sequence. Table 5.1 shows the accuracy of matching each predicted value with the corresponding value in ground truth for different slots (i.e., sub-module matching). Table 7 shows the accuracy of matching the list of predicted values in a SQL query with the list of values in ground truth SQL (i.e., strict slot matching). It is obvious that the current decoders lack the ability in parsing a long sequence of conditions. A stronger decoder needs to be designed for this task.

Table 7: Strict slot accuracy on unique conditions. It matches the list of predicted slot value from sub-modules with corresponding list in ground truth. Since the output $C_{\#}$ is a number, the strict slot accuracy for slot $C_{\#}$ is same as sub-module accuracy.

Type	Set	$C_{\#}$	C_n	C_o	C_v	C_c	C_p
Boolean-Type	Dev	56.3	35.4	50.0	56.3	54.2	52.1
	Test	70.4	15.5	53.5	70.4	67.6	64.8
Order-Sensitive	Dev	43.5	6.5	0.7	43.5	7.3	2.2
	Test	24.5	1.9	0	24.5	7.5	1.9
Counting-Based	Dev	0	0	0	0	0	0
	Test	0	0	0	0	0	0

6. Conclusion

In this paper, we present a new dataset for the eligibility criteria-to-sql task named Criteria2SQL and a strong baseline model for the task. The eligibility criteria are collected from real clinical trials, which makes the dataset represent real-world challenges for criteria-to-sql problems. A context-free grammar and corresponding slots are designed for our new sketch-based neural semantic parser to predict and generate complex conditions for the order-sensitive, counting-based, and Boolean-type eligibility criteria. A baseline model is proposed for the criteria-to-sql task that outperforms state-of-art-models. The experiment results show the task is very challenging especially the column name prediction, priority prediction and prediction of a long sequence of conditions, and leaves a lot of research opportunities for the researchers to investigate and improve.

7. Bibliographical References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 33–43.

Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., and Zhang, D. (2019). Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.

Hayati, S. A., Olivier, R., Avvaru, P., Yin, P., Tomasic, A., and Neubig, G. (2018). Retrieval-based neural code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 925–930.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hripcsak, G., Duke, J. D., Shah, N. H., Reich, C. G., Huser, V., Schuemie, M. J., Suchard, M. A., Park, R. W., Wong, I. C. K., Rijnbeek, P. R., et al. (2015). Observational health data sciences and informatics (ohdsi): opportunities for observational researchers. *Studies in health technology and informatics*, 216:574.

Hwang, W., Yim, J., Park, S., and Seo, M. (2019). A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Iyer, S., Konstas, I., Cheung, A., and Zettlemoyer, L. (2018). Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652.

Kang, T., Zhang, S., Tang, Y., Hruby, G. W., Rusanov, A., Elhadad, N., and Weng, C. (2017). Eliie: An open-source information extraction system for clinical trial eligibility criteria. *Journal of the American Medical Informatics Association*, 24(6):1062–1071.

Kushman, N., Artzi, Y., Zettlemoyer, L., and Barzilay, R. (2014). Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281.

Le, P. and Zuidema, W. (2012). Learning compositional semantics for open domain semantic parsing. *Proceedings of COLING 2012*, pages 1535–1552.

Levy-fix, G., Yaman, A., and Weng, C. (2015). Structuring clinical trial eligibility criteria with the common data model.

Lin, K., Bogin, B., Neumann, M., Berant, J., and Gardner, M. (2019). Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Lonsdale, D. W., Tustison, C., Parker, C. G., and Embley, D. W. (2008). Assessing clinical trial eligibility with logic expression queries. *Data & Knowledge Engineering*, 66(1):3–17.

Luo, Z., Johnson, S. B., Lai, A. M., and Weng, C. (2011). Extracting temporal constraints from clinical research eligibility criteria using conditional random fields. In *AMIA annual symposium proceedings*, volume 2011, page 843. American Medical Informatics Association.

Mehta, P., Mishra, P., Athavale, V., Shrivastava, M., and Sharma, D. (2017). Deep neural network based system for solving arithmetic word problems. In *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 65–68.

Milian, K. and ten Teije, A. (2013). Towards automatic patient eligibility assessment: from free-text criteria to queries. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 78–83. Springer.

Pasupat, P. and Liang, P. (2015). Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1470–1480.

Reddy, S., Täckström, O., Collins, M., Kwiatkowski, T., Das, D., Steedman, M., and Lapata, M. (2016). Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Ross, J., Tu, S., Carini, S., and Sim, I. (2010). Analysis of eligibility criteria complexity in clinical trials. *Summit on Translational Bioinformatics*, 2010:46.

Tu, S. W., Peleg, M., Carini, S., Bobak, M., Ross, J., Rubin, D., and Sim, I. (2011). A practical method for transforming free-text eligibility criteria into computable criteria. *Journal of biomedical informatics*, 44(2):239–250.

Wang, P., Shi, T., and Reddy, C. K. (2019). A translate-edit model for natural language question to sql query generation on multi-relational healthcare data. *arXiv preprint arXiv:1908.01839*.

Xu, X., Liu, C., and Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yin, P., Lu, Z., Li, H., and Kao, B. (2016). Neural enquirer: learning to query tables in natural language. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2308–2314. AAAI Press.

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., and Radev, D. (2018a). Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., et al. (2018b). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Yuan, C., Ryan, P. B., Ta, C., Guo, Y., Li, Z., Hardin, J., Makadia, R., Jin, P., Shang, N., Kang, T., et al. (2019). Criteria2query: a natural language interface to clinical databases for cohort definition. *Journal of the American Medical Informatics Association*, 26(4):294–305.

Zhong, V., Xiong, C., and Socher, R. (2017).

Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.